An Investigation of Inter-Domain Control Aggregation Procedures

Rute Sofia
Roch Guérin
Pedro Veiga

DI–FCUL TR–2002–14

September 2002

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1749–016 Lisboa
Portugal

Technical reports are available at `http://www.di.fc.ul.pt/tech-reports`. The files are stored in PDF, with the report number as filename. Alternatively, reports are available by post from the above address.

An Investigation of Inter-Domain Control Aggregation Procedures*

Rute Sofia(1) Roch Guérin(2) Pedro Veiga(3)

Abstract

Current Quality of Service models such as those embodied in the Differentiated Services proposals, rely on data path aggregation to achieve scalability. Data path aggregation bundles into a single aggregate multiple individual flows with the same quality requirements, hence decreasing the amount of state that needs to be kept along a path. A similar scalability concern exists on the control path, where the state required to account for individual reservations needs to be minimized. There have been a number of proposals aimed at control path aggregation, and the goal of this report is to expand on these works in an attempt to gain a better understanding of the various parameters that influence the efficiency of different approaches. In particular, we focus on inter-domain control aggregation, and compare an Autonomous System (AS) sink-tree based approach with several examples of a shared AS segment based approach. The comparison is done in terms of the amount of state that is kept, both within a given AS, as well as at edge routers. The comparison is carried out primarily through simulations, but we also develop a simple analytical model for a basic AS configuration, which provides additional insight into the impact of different parameters on the efficiency of each approach. Our main contributions are in providing a greater understanding into the design of efficient control path aggregation methods.

I. INTRODUCTION

Data path *Quality of Service* (QoS) issues are by now reasonably well understood, and a number of different alternatives have been proposed and investigated, e.g., *Integrated Services* (*IntServ*) [3] and *Differentiated Services* (*DiffServ*) [11], each representing a different trade-off in terms of capability and scalability. However, the same understanding is not really available when it comes to control path issues. The control path consists primarily of mechanisms for reserving and maintaining the necessary data path resources, as embodied in proposals such as *Internet Streaming Protocol (ST-II)* [5] and *Resource Reservation Protocol (RSVP)* [4], with the latter being the current solution of choice for most new IP services. The main concern with these proposals is their scalability, specially when thinking of inter-domain links that are expected to carry a large volume of individual reservation requests.

Our main motivation is, therefore, to gain a better perspective into the scalability of various control mechanisms, and their ability to handle large reservation volumes. We focus on inter-domain control reservations, as we expect them to be the most stressful in terms of scalability. Our approach is not so much to propose a specific mechanism, but instead to try to gain a basic understanding of factors and parameters that affect the scalability of inter-domain control reservation mechanisms. In particular, we focus on evaluating various *aggregation* techniques that attempt to minimize the amount of state and processing due to resource reservation on inter-domain links. Some of the basic questions involved are how, when and where to aggregate individual reservation requests. To gather information about a path and since we are considering inter-domain aggregation, we rely on substantial information provided by the *Border Gateway Protocol* (BGP) [16], the current dynamic solution for inter-domain information exchange.

There are several possible criteria that can be used to decide how to aggregate reservation requests on links connecting different routing domains or *Autonomous Systems* (AS's). Aggregation can, for example, be done on the basis of a single shared AS hop, or on the basis of a shared AS path segment, or simply be based on having the same destination AS, as proposed in [6]. These different options translate into different amounts of state being maintained at different locations in the network. In general, state needs to be kept for each individual reservation at all aggregation and deaggregation points, while state is kept for aggregate reservations at all the intermediate inter-domain links they traverse. Hence, the goal of a scalable solution is to minimize the overall amount of reservation state in the network, as well as the amount of reservation state that any router needs to maintain.

*This work was developed at University of Pennsylvania, ESE Department, Networking and Multimedia Lab, and was supported by the program Programa Operacional Sociedade de Informação (POSI), of the Portuguese Fundação para a Ciência e Tecnologia and of the European Union FEDER. Scholarship reference: PraxisXXI/BD/18246/98, and by NSF grants ANI-9902943, ANI-9906855, and ITR-0085930.

(1) rsofia@seas.upenn.edu, ESE, University of Pennsylvania, Philadelphia, PA 19104-6390 and Department of Informatics, University of Lisbon.
(2) guerin@ee.upenn.edu, ESE, University of Pennsylvania.
(3)pmv@di.fc.ul.pt, Department of Informatics, University of Lisbon.

In addition to the amount of reservation state needed, a scalable solution should also take into account processing and signaling requirements, ensuring that both are kept as low as possible. A related factor is the bandwidth efficiency of a solution, and in particular how often the bandwidth allocated to an aggregate reservation is updated. Ideally, bandwidth should be updated after every change to the individual reservations of an aggregate. This would ensure that only the minimum possible amount would be allocated, but most likely translating into a significant signaling load. Alternatively, bandwidth allocation could be updated less frequently to minimize signaling overhead. However, this could affect network efficiency by providing some aggregate reservations with more bandwidth than they really need, potentially preventing others from getting the bandwidth they require.

All of the above represent issues that need to be explored, and carrying out such a comprehensive investigation is clearly beyond the scope of a single document. In this report, we concentrate on the aspect of state optimization and consider two representative families of possible algorithms. The first one makes aggregation decisions on the basis of shared AS sink-trees, while the second relies on shared AS path segments. We consider algorithms that belong to each family, and evaluate their cost in terms of the amount of state they require both at the AS level and at edge routers.

The remainder of the report is organized as follows: Section 2 covers related work. Section 3 presents control aggregation issues and definitions. Section 4 describes the two aggregation approaches under consideration, and presents a simple analytical model for computing the amount of state required by the several candidate algorithms. Section 4 is devoted to the evaluation of the performance of the algorithms by means of simulations. Finally, Section 5 summarizes findings and outlines future work.

II. Related Work

Guérin et al. [8] present a survey of possible approaches to aggregate RSVP requests assuming unicast scenarios and covering issues such as RSVP state management and path characterization. The survey proposes the use of aggregation *tunnels*, i.e., pipes between entry and exit points of a defined *aggregation region*, i.e., a cloud of routers where regular RSVP messages are ignored. A similar approach is followed by Berson et al. [10]. They consider unicast and multicast scenarios, focusing also on RSVP aggregation within an aggregation region. These two approaches are concerned with RSVP scalability: RSVP requires all the routers on the path of an individual reservation to maintain state dedicated to that reservation. The resulting amount of state can be overwhelming especially for backbone routers that may have to support a large number of simultaneous requests. The two proposals reduce the amount of state by aggregating individual requests inside an aggregation region. However, in both proposals, an aggregation region is typically synonymous with an Autonomous System, i.e., ingress and egress routers are entry and exit points for the AS, respectively. As a result, neither considers the problem of inter-domain control aggregation, which is the focus of this paper.

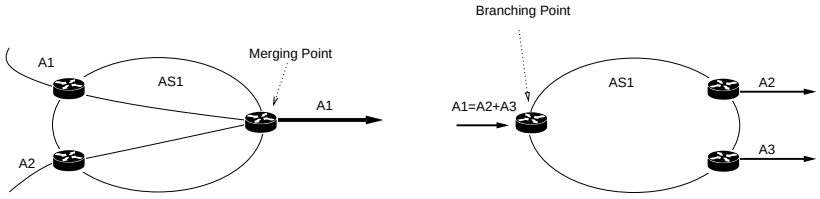
Pan et al. [6] was the first work to explicitly consider the problem of inter-domain aggregation, for which it introduced an inter-domain signaling protocol, the *Border Gateway Reservation Protocol (BGRP)*. BGRP aggregates control information by merging requests that have the same destination *AS*. For each reservation, BGRP sends a pair of control messages along the path that an aggregate will follow to reach its destination according to BGP rules, i.e., a *sink-tree*. On each AS along the path, edge routers keep information regarding the tree each individual reservation belongs to, its sink or root, and the amount of bandwidth to reserve for the tree. Using a tree has the advantage of avoiding *intermediate deaggregation points*, i.e., AS's between source and destination where individual reservations need to be regenerated. Hence, deaggregation takes place only at the destination AS. Pan et al. show that BGRP has good performance when compared with RSVP without aggregation. However, BGRP was not compared to other possible aggregation methods. Hence, assessing its effectiveness as an inter-domain solution remains an open question. Answering such a question and exploring the space of possible approaches is one of the motivations of our work.

III. Inter-domain Control aggregation definitions and Terminology

In this section, we introduce some terminology and concepts related with inter-domain control aggregation that will be used in the next sections. Throughout the document, we consider that an *aggregation region* or *aggregation domain* is synonymous with an AS. An *ingress router* is a router placed at the boundary (*edge*) of an AS, crossed by traffic that enters the AS. Similarly, an *egress router* is a router placed at the boundary of an AS, but crossed by traffic that exits the AS.

Requests having in common some path characteristics and crossing the same egress router can be bundled together in an *aggregate*. For instance, requests going to the same destination can be aggregated together hence being treated as a single

request by edge routers along a path. An aggregate is named *originating* if it starts in the current AS; it is named *ending* if it ends in the current AS, having therefore to be deaggregated; it is named *transient* if it is just passing by the current AS. Consequently, aggregates are characterized by their starting and ending AS's. An *aggregator* is a process in charge of processing and possibly merging requests as they leave an AS, hence positioned at egress routers. A *deaggregator* is a process in charge of splitting ending aggregates into requests, hence positioned at ingress routers. *Merging* of aggregates takes place when aggregates that cross different ingress routers and the same egress router at an AS, have the same aggregation requisites (for instance, share a path segment). Such is the case exemplified in Fig.1 (a), where aggregate A2 is merged into aggregate A1. A *branching point* is an ingress router where arriving aggregated information will be split, due to the necessity of having to follow different paths, thus crossing different egress routers, as exemplified in Fig. 1 (b). *Intra-domain* links are connections between networking elements inside an AS, while *inter-domain* links are connections between neighboring ASs. An *AS hop* represents an inter-domain hop. An AS j is *downstream* of an AS i if it is between i and a destination AS; AS j is *upstream* of AS i if it is between a source AS and i . *First-level* aggregation, happens when an aggregate bundles together individual requests. *Multi-level* aggregation occurs when an aggregate of level l contains at least an aggregate of level $l-1$.



(a) Merging point (b) Branching point

Fig. 1. (a) Merging Point: aggregates A1 and A2 entering AS1 through different ingress points will be merged together, forming a single aggregate, A3; (b) Branching point: aggregate A1 entering AS1 will have to be split into aggregates A2 and A3, since these aggregates will follow different paths

Let us now consider the network scenario illustrated in Fig. 2, where each circle represents an AS, and A's and D's (placed at edge routers) represent aggregators and deaggregators, respectively. To simplify visualization, let us suppose that traffic flows only from left to right. In reality, each edge router can be seen as both an ingress and egress point, since links are bidirectional. Core routers and intra-domain signaling mechanisms are ignored, since this study is about inter-domain aggregation.

Let us also consider that A in AS 1 receives reservation requests from two sources, Src_1 and Src_2, which have destinations Dst_1 and Dst_2 in AS 5, respectively. Router A in AS 1 decides to aggregate these two requests based on their common destination AS. Thus, A will send an aggregate (A1) to the corresponding deaggregator D in AS 5 (solid line arrows). In terms of resources requested along the path, this aggregate represents the sum of the bandwidth of each individual request; in terms of state it can be seen as a single request.

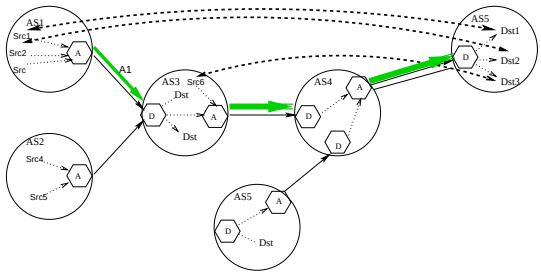


Fig. 2. Generic model representation

If the request is accepted, edge routers along the path maintain information about A1. Also, along the AS path, there might be more requests merged into this transient aggregate. Such is the case of the request sent by Src_6 in AS 3 to Dst_3 in AS 5.

Merging of requests results in lowering the state needed to be kept along the path: the built aggregate is transmitted between the first aggregator and the last deaggregator, thus avoiding to keep information about individual requests in intermediate routers.

State represents information about reservations that routers along a path need to store. Hence, from a router perspective, state is associated with the interfaces crossed by the requests, as illustrated in Fig. 3.

We consider that a request, whether individual or aggregate, occupies one unit of state: if x requests are mapped into 1 aggregate at an A, corresponding state is $x + 1$ units; when an aggregate that contains x requests is deaggregated at a D, the state occupied is $1 + x$; if an aggregate is transient, it requires 2 units of state per A and per D.

Hence, the average reservation state,Q, for an edge router i in an AS is given by Eq. 1 , where I_i represents state due to incoming reservations and O_i due to outgoing reservations.

Fig. 3. State at an edge router

$$Q_i = I_i + O_i \tag{1}$$

Correspondingly, the average state for an AS m, S_m, (Eq. 2) is simply obtained by summing the state of its n edge routers.

$$S_m = \sum_{i=0}^{n} Q_i \tag{2}$$

Both S_m and Q_i are relevant performance measures for a given aggregation scheme.

Tracking state at the AS level gives an overall measure of performance, while tracking it at the router level can help identify variations in state that routers are required to maintain. For example, an aggregation scheme could achieve a low AS level state quantity by having state concentrated at a few routers.

In the next section, we use a simple analytical model to explain state accounting for the two aggregation approaches and the several aggregation algorithms, derived from these approaches.

IV. Aggregation Approaches and algorithms

To describe and compare the aggregation approaches, we use the generic aggregation scenario illustrated in Fig. 4 , where AS's 1 to P represent source AS's and $P + K + 1$ to $P + K + N$ destination AS's. Between source and destination AS's, there is a segment with K AS's. RI represents an edge router in charge of ingress operations, i.e., possible deaggregation. RE represents an edge router in charge of egress operations, i.e., aggregation.

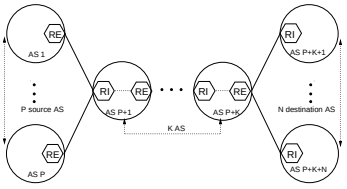


Fig. 4. Inter-Domain Aggregation Scenario

For ease of understanding and visualization, traffic flows only from left to right. We assume that each of the source AS's wants to establish Y individual reservations with each of the destination AS's and so, the average amount of requests is $Y * P * N$. In the selected topology, all paths have a size of $K + 1$ AS hops, where K is a variable that can be set to reflect a typical AS hop count, e.g., based on a given AS path size distribution. To obtain realistic values for K, we use the values collected by Telstra [12] and presented in [9]. This data is based on BGP measurements obtained from five major operators in 2001 and gathered from a total of 60978 AS. Among other facts, it shows that the current maximum AS path has a size of 10 AS's. So, K is less than or equal to nine AS's, since the biggest path in our scenario has $K + 1$ AS's. This model, an AS-level *dumbbell* [2] topology, is simple and sufficient to explain the state accounting methodology we use, since state along any path differs as a function of an AS location: sources, destinations, and intermediate AS's.

In our model, source AS's (1 to P) keep only state related with outgoing reservations. Destination AS's ($P + K + 1$ to $P + K + N$) keep only state due to incoming reservations. State accounting for AS's $P + 1$ to $P + K$ is more complex and depends on the aggregation approach used.

A. Sink-Tree AS Based Approach

Fig. 5 displays an example of state accounting for the scenario of Fig. 4 when using a sink-tree AS based aggregation approach, as proposed in BGRP. Requests are aggregated on the basis of their destination AS, so that the resulting aggregate is in the form of a sink-tree whose root is the destination AS. In other words, all requests with a common destination AS are mapped onto the same tree, independently of their source AS.

Since each source AS is generating Y individual reservations for each destination AS, we have $Y * N$ individual requests per source AS. Also, each source AS creates N aggregates, because aggregation is based on destination AS's. Therefore, there is a total of $P * N$ aggregates entering AS $P + 1$. At this AS, merging of the $P * N$ aggregates takes place based on their respective destination AS, which results in a total of N outgoing aggregates. Deaggregation occurs only at destination AS's, where incoming aggregates are deaggregated into individual reservation requests.

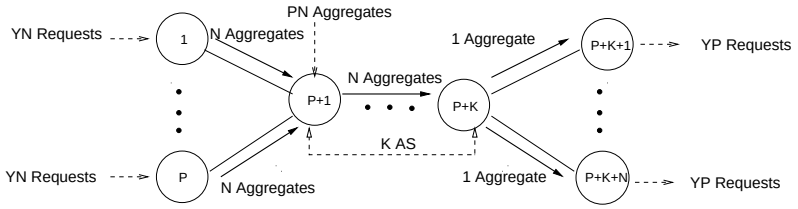


Fig. 5. Example of accounting for the sink-tree approach

Tab. I details state kept in each AS, showing units both ingress and egress routers, and Eq. 3 gives the global state count for a sink-tree aggregation approach in this particular scenario.

$$S_1 = PN(2Y + 4) + 4NK - 2N \tag{3}$$

TABLE I
GLOBAL STATE FOR THE SINK-TREE APPROACH

AS	1	(...)	P	P+1	P+2	(...)	P+K	P+K+1	(...)	P+K+N
Ingress (IN,OUT)	-	-	-	NP, NP	N, N	N, N	N, N	1, YP	1, YP	1, YP
Egress(IN,OUT)	YN, N	YN, N	YN,N	NP, N	N, N	N, N	N, N	-	-	-
Total	(Y+1)N	(Y+1)N	(Y+1)N	3NP+N	4N	4N	4N	YP+1	YP+1	YP+1

B. Shared AS Segment Based Approach

In this approach, aggregation decisions are made based on the existence of a shared AS path segment between an existing aggregate and a new reservation. In contrast, the sink-tree approach requires a shared segment that extends all the way to the destination AS. In the shared segment approach reservation requests can be assigned to any aggregate with an ending point upstream of their destination AS. If no such aggregate exists, a new one is created, not necessarily extending all the way to the destination AS. The motivation for such flexibility is that shorter aggregates may accommodate more easily additional future requests. On the one hand, by aggregating reservation requests that share only a path segment, we expect to minimize the number of aggregates in use and hence, global state. On the other hand, this process can result in having multiple deaggregation points, each contributing with state, wiping out the advantage of reducing the number of aggregates. In contrast, in the sink-tree approach individual requests require only one deaggregation point at the destination AS that rooted the sink-tree. Our goal is to explore if proper selection of the shared segment size can lead to a solution with better performance than a sink-tree.

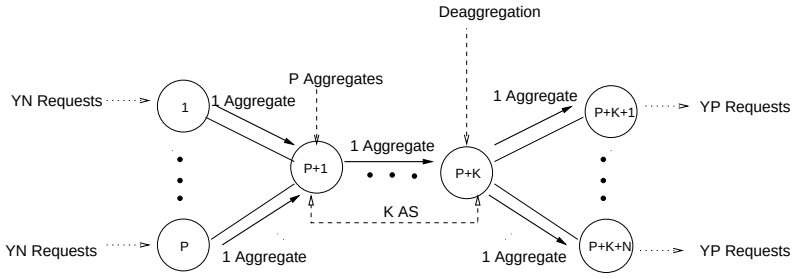


Fig. 6. Example of accounting for the shared segment approach

Fig. 6 exemplifies accounting for the scenario illustrated in Fig. 4 (a), when AS $P+K$ is the chosen deaggregation point. Note that this is the obvious choice in this particular example, but that more complex configurations may not yield such

a clear choice. Tab. II shows the amount of state maintained in each AS when using the shared segment approach, while Eq. 4 gives global state, S_2.

TABLE II
GLOBAL STATE FOR THE SHARED SEGMENT APPROACH

AS	1	(...)	P	P+1	P+2	(...)	P+K	P+K+1	(...)	P+K+N
Ingress (IN,OUT)	-	-	-	P,P	1,1	1,1	1,YPN	1, YP	1, YP	1, YP
Egress(IN,OUT)	YN, 1	YN, 1	YN,1	P,1	1,1	1,1	YPN, N	-	-	-
Total	YN+1	YN+1	YN+1	3P+1	4	4	2YPN+N+1	YP+1	YP+1	YP+1

$$S_2 = 4YPN + 4P + 2N + 4K - 6 \tag{4}$$

Comparing Eq. 3 and Eq. 4 by varying the number of sources, destinations and also the value of K, we can see that the performance of both the sink-tree and the shared segment approaches experiences variations. We also see that the number of individual requests per source AS, Y, has more impact on state equation S_2 than on S_1. This means that the shared segment approach is likely to be more sensitive to the intensity of individual requests than the sink-tree approach.

In order to understand possible variations and also explore how to choose an optimal deaggregation point, we introduce next two algorithms based on the shared segment approach, and compare them with a sink-tree based algorithm, namely, BGRP.

Biggest Possible Shared Segment (BPS): This algorithm is triggered each time a new reservation request arrives. It then checks whether or not an adequate aggregate exists. By adequate we mean that the aggregate has to have in common with the request a segment with a pre-determined size. For instance, if a request has only one AS hop in common with the aggregate, then aggregating them might bring only a small advantage. In case there are multiple possible aggregates, BPS looks for the one with the best size, i.e., a size that minimizes the cost of deaggregation.

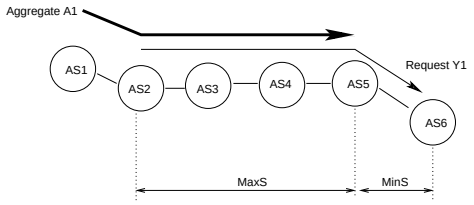


Fig. 7. Shared segment path size for BPS

Choosing an aggregate is, therefore, made based on the Maximum Shared Segment (MaxS) shared with the request. It has also to take into consideration the Minimum Surplus Segment (MinS), which corresponds to the remaining path segment from the end point of the chosen aggregate to the destination AS of the request. Fig. 7 displays a diagram with MaxS and MinS segments. It illustrates an individual request, Y1, originating in AS 2 and ending in AS 6, that has a possible aggregate candidate A1. A1 starts in AS 1 and ends in AS 5, three hops ahead of AS 2. MaxS is the segment between AS 2 and AS 5. MinS is the segment from AS 5 to AS 6.

TABLE III
VALUES OF MAXS, MINS

Request Path Size (RPS)	< 5	>= 5, <= 10	> 10
MaxS	$0.8 * RPS$	$0.7 * RPS$	$0.6 * RPS$
MinS	$0.2 * RPS$	$0.3 * RPS$	$0.4 * RPS$

Tab. III displays values for MaxS and MinS, where $[x]$ represents the closest integer to x. These values were chosen having in mind a typical path size based on the path distribution mentioned in Section IV. If there is no aggregate with the required characteristics, a new one is created with a size equal to MaxS.

Tab. IV details state kept per AS when using BPS in the context of the topology of Fig. 4. The placement of the intermediate deaggregation point,AS D, is influenced by the size of the segment between source and destination AS's. Hence, the value of K is relevant to total state. For our model, Eq. 5 gives the position of AS D.

$$D = \begin{cases} P + \frac{K}{2} + 1, \; K+1 < 5 \\ P + \frac{K+1}{2} + 2, \; 5 \leq K+1 \leq 10 \end{cases} \tag{5}$$

TABLE IV
BPS STATE PER AS

AS	1	(...)	P	P+1	(...)	Deaggregator, D	(...)	P+K+1	(...)	P+K+N
Ingress (IN,OUT)	-	-	-	P,P	1,1	1,YPN	N, N	1, YP	1, YP	1, YP
Egress(IN,OUT)	YN, 1	YN, 1	N,1	P,1	1,1	YPN,N	N, N	-	-	-
Total	YN+1	YN+1	N+1	3P+1	4	2YPN+N+1	4*N	YP+1	YP+1	YP+1

According to Eq. 5, there are only two deaggregation locations: AS D and the destination. But as indicated in Eq. 6, the total amount of state S_3 varies based on the position of D relative to the destination AS's.

$$S_3 = \begin{cases} 4YPN + 4P - 2N + 2NK + 2K - 2, \; K+1 < 5 \\ 4YPN + 4P - 8N + 2NK + 2K + 4, \; 5 \leq K+1 \leq 10 \end{cases} \tag{6}$$

BPS has the advantage of choosing deaggregation points that take into account the path size of requests.

Hence, it is sensitive to the order and characteristics of requests that trigger the establishment of aggregates. For example, if the first requests arriving to an aggregator have small path sizes, the corresponding initial aggregates will tend to have small path sizes also, which will lead to a small number of aggregates, but possibly to many deaggregation points. However, if initial requests have large path sizes, then the path size of initial aggregates will also be large. This implies the use of less deaggregation points, but possibly ending up with a larger number of aggregates.

Segments with 'Weighted' Deaggregation Point (WDS): This algorithm intends to remedy the deficiency of BPS in limiting aggregate path sizes due to the path size of the first requests received, by including information on the likelihood that a given AS will be a termination point for many future requests. Specifically, WDS assumes that AS's with a larger number of downstream neighbor AS's are more likely to be deaggregation points. This makes such AS's better candidates for being the end-point of an aggregate, and is combined with the distance from the aggregation point when deciding how to create new aggregates. In other words, the aggregator computes a weight, W_m, for each AS m of each path request, based on the number of downstream AS neighbors and the distance from the aggregator to AS m. It then chooses as deaggregation point the AS with the biggest weight. Eq. 7 defines W_m, where n_j represents a downstream neighbor of m and d represents the distance from the origin AS to m, given in AS hops:

$$W_m = \sum_j n_j * d, \forall j, m \tag{7}$$

There are two special cases for the algorithm. The first occurs when two AS's yield the same weight value. In this case, the algorithm chooses the AS nearest to the destination. The second occurs when the destination AS is a *leaf*, i.e., it has no downstream neighbors. For this case, the algorithm assumes that $n_j = 1$.

Tab. VI displays the amount of state kept per AS in the scenario of Fig. 4, when using WDS. In this scenario, AS K is always selected as the deaggregation point, since it yields the largest weight. Total state for WDS, S_4, is given in Eq. 8.

$$S_4 = 4YPN + 4P + 2N + 4K - 6 \tag{8}$$

TABLE V
WDS STATE PER AS

AS	1	(...)	P	P+1	P+2	(...)	P+K	P+K+1	(...)	P+K+N
Ingress (IN,OUT)	-	-	-	P,P	1,1	1,1	1,YPN	1, YP	1, YP	1, YP
Egress(IN,OUT)	YN, 1	YN, 1	YN,1	P,1	1,1	1,1	YPN, N	-	-	-
Total	YN+1	YN+1	YN+1	3P+1	4	4	2YPN+N+1	YP+1	YP+1	YP+1

The major drawback of WDS is that it has to know beforehand the number of downstream neighbors for each AS. This information has to be kept and updated at each aggregator. Nevertheless, WDS has the advantage of choosing deaggregation points that are less sensitive to the characteristics of individual requests and the order in which they are received.

C. Multi-Level Segment Aggregation with 'Weighted' Deaggregation Point (MLWDS)

The two algorithms based on the shared segment approach seem to reduce the number of aggregates created when compared with the sink-tree approach. However, they introduce the cost of having to keep information about requests mapped to an aggregate at locations upstream of destination AS's. This cost might increase significantly global state and hence, the reduction of aggregates provided by the use of a shared segment approach might not be enough to achieve an optimal aggregation method. Therefore, the problem resides in having to keep information about individual individual requests at intermediate deaggregation locations. This cost can be avoided if instead of performing first-level aggregation, an algorithm performs multi-level aggregation: intermediate locations won't need to keep information about individual requests, only about aggregates.

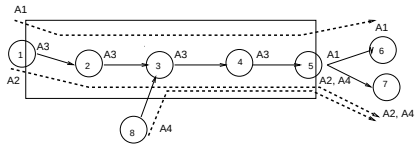


Fig. 8. Aggregation with MLWDS

MLWDS is an algorithm that performs second-level aggregation at source AS's, by first creating in each source an aggregate to the AS destination and then aggregating it into a second-level aggregate, created according to the algorithm WDS, i.e., having as destination an AS upstream of the AS destination of the request. MLWDS behavior is illustrated in Fig. 8, where dashed lines represent first level aggregates. In AS 1, there are two sources that want to establish reservations with destinations in AS's 6 and 7. Then, MLWDS creates $A1$ and $A2$, first-level aggregates that end in AS's 6 and 7, respectively. Then, MLWDS creates a second-level aggregate, $A3$, at AS 1, following WDS rules. Hence, $A3$ destination will be AS 5. $A3$, represented by the rectangle between AS's 1 and 5, is a second-level aggregate unto which A1 and A2 are mapped. Therefore, between AS 1 and 5 edge routers keep only state due to $A3$. AS 5 keeps information about $A3$ at ingress and about $A1$ and A2 at egress, since these aggregates have as destination AS's downstream of AS 5. It should be noticed that in this example, aggregates A1 and A2 will not be reaggregated. However, in more complex scenarios, it is most likely that multi-level aggregation will occur at locations upstream of the destination AS of the request.

Considering that AS 8 also wants to establish reservations with AS 7, then it will have to create an aggregate, $A4$, having as destination AS 8. When reaching AS 3, a merging point, $A4$ will be reaggregated with $A3$. Hence, AS 4 will only keep information regarding $A3$.

The idea behind MLWDS is that it is most likely that a source AS has more than one request destined to the same destination AS. However, since the shared-segment approach minimizes the number of aggregates created, MLWDS will create a second-level aggregation, to avoid the cost of deaggregating to the level of individual requests in intermediate requests.

There is an exception in the behavior of MLWDS: if, according to the aggregation rules of WDS, the best deaggregation location of an aggregate is the destination AS, then MLWDS won't perform second-level aggregation for that aggregate.

Considering again Fig. 4, Tab. IV details state kept per AS when using MLWDS. Global state for MLWDS, S_5, is given by Eq. 9.

$$S_5 = 2YPN + 4P + 3PN + 2N + 4K - 6 \tag{9}$$

In Tab. IV, we can see that state due to individual requests is now only kept at sources and destinations.

TABLE VI
MLWDS STATE PER AS

AS	1	(...)	P	P+1	P+2	(...)	P+K	P+K+1	(...)	P+K+N
Ingress (IN,OUT)	-	-	-	P,P	1,1	1,1	1,PN	1, YP	1, YP	1, YP
Egress(IN,OUT)	YN, N+1	YN, N+1	YN,N+1	P,1	1,1	1,1	PN, N	-	-	-
Total	YN+N+1	YN+N+1	YN+N+1	3P+1	4	4	2PN+N+1	YP+1	YP+1	YP+1

In the next section, we give a brief comparison of the four algorithms for the scenario presented.

D. A Comparison Example

In this section we present a simple comparison of BGRP, BPS, WDS, and MLWDS in terms of the state they require for the configuration of Fig. 4. We aim to show changes in state due to the variation of the number of sources, destinations,

and the size of the segment between sources and destinations, K. Hence, K is taken to be between 1 and 10, while P, N, and Y are taken to be between 1 and B, a large quantity. The possible combinations of P, N, Y, and K in terms of the maximum and the minimum values of Eq. 3, Eq. 6, and Eq. 8 are displayed by rows in Tab. VII.

TABLE VII
GLOBAL STATE COMPARISON, $1 < K < 10$

ROW	P	N	Y	K	BGRP	BPS	WDS	MLWDS
1	1	1	1	1	$S_1 = 8$	$S_3 = 8$	$S_4 = 8$	$S_5 = 9$
				10	$S_1 = 44$	$S_3 = 44$	$S_4 = 44$	$S_5 = 45$
2	1	1	B	1	$S_1 \approx 2B$	$S_3 \approx 4B$	$S_4 \approx 4B$	$S_5 \approx 2B$
				10	$S_1 \approx 2B$	$S_3 \approx 4B$	$S_4 \approx 4B$	$S_5 \approx 2B$
3	1	B	1	1	$S_1 \approx 8B$	$S_3 \approx 4B$	$S_4 \approx 6B$	$S_5 \approx 7B$
				10	$S_1 \approx 44B$	$S_3 \approx 16B$	$S_4 \approx 6B$	$S_5 \approx 7B$
4	1	B	B	1	$S_1 \approx 2B^2 + 6B$	$S_3 \approx 4B^2$	$S_4 \approx 4B^2 + 2B$	$S_5 \approx 2B^2 + 5B$
				10	$S_1 \approx 2B^2 + 42B$	$S_3 \approx 4B^2 + 12B$	$S_4 \approx 4B^2 + 2B$	$S_5 \approx 2B^2 + 5B$
5	B	1	1	1	$S_1 \approx 6B$	$S_3 \approx 8B$	$S_4 \approx 8B$	$S_5 \approx 9B$
				10	$S_1 \approx 6B$	$S_3 \approx 8B$	$S_4 \approx 8B$	$S_5 \approx 9B$
6	B	1	B	1	$S_1 \approx 2B^2 + 4B$	$S_3 \approx 4B^2 + 4B$	$S_4 \approx 4B^2 + 4B$	$S_5 \approx 2B^2 + 7B$
				10	$S_1 \approx 2B^2 + 4B$	$S_3 \approx 4B^2 + 4B$	$S_4 \approx 4B^2 + 4B$	$S_5 \approx 2B^2 + 7B$
7	B	B	1	1	$S_1 \approx 6B^2 + 2B$	$S_3 \approx 4B^2 + 4B$	$S_4 \approx 4B^2 + 6B$	$S_5 \approx 7B^2 + 6B$
				10	$S_1 \approx 6B^2 + 38B$	$S_3 \approx 4B^2 + 16B$	$S_4 \approx 4B^2 + 6B$	$S_5 \approx 7B^2 + 6B$
8	B	B	B	1	$S_1 \approx 2B^3 + 4B^2 + 2B$	$S_3 \approx 4B^3 + 4B$	$S_4 \approx 4B^3 + 6B$	$S_5 \approx 2B^3 + 3B^2 + 6B$
				10	$S_1 \approx 2B^3 + 4B^2 + 38B$	$S_3 \approx 4B^3 + 16B$	$S_4 \approx 4B^3 + 6B$	$S_5 \approx 2B^3 + 3B^2 + 6B$

Row one represents a configuration with one source and destination AS, as well as one request only. This is a very simple configuration, just used to exemplify the behavior of the algorithms for configurations with a small number of source and destination AS's, as well as low intensity of requests. For this case, the four algorithms show similar performance, which means that none of the varied parameters, in small quantity, has significant impact on the state the algorithms require. When the number of requests is considerably increased (row two), both MLWDS and BGRP require less state than the other two algorithms: this happens because MLWDS and BGRP keep state of individual reservations only at source and destination AS's, while BPS and WDS add to this the cost of having to keep information about individual requests at one intermediate deaggregation point. When K changes from 1 to 10, the amount of state for any of the algorithms remains constant, which shows that the value of K in this particular configuration does not influence significantly the average global state required by any of the algorithms.

If we keep the number of sources low, but increase the number of destinations (row three), BGRP presents the worst performance of the four algorithms. This happens, because BGRP state depends on the number of destination AS's. Also, in this scenario, WDS and MLWDS performance is approximate, because they both choose an intermediate deaggregation point and the number of requests is small. But, if we increase again the number of reservation requests (row four), then MLWDS performs better than the other three algorithms. WDS and BPS show similar performance, which is worse than the performance of either BGRP or MLWDS: in the current configuration, WDS and BPS have to keep close to twice the amount of state needed by BGRP or MLWDS.

In row five, we change the configuration by introducing a large number of sources and a small number of destinations, as well as low intensity of requests. The four algorithms show close performance, even though BGRP is the algorithm requiring less state. The value of K is not significant under this configuration, since whether K is equal to 1 or 10, the value of global state remains approximately the same. When increasing the intensity of requests (row six), WDS and BPS performance decays: they need to keep close to the double of the amount of state of BGRP or MLWDS. Once again, state required for any of the algorithms does not depend on the value of K, since whether we equal K to 1 or to 10, state remains the same.

In a configuration with a large number of sources and destinations but small number of requests (row seven), both MLWDS and BGRP require more state than either BPS or WDS, because deaggregation is not optimized along the segment between sources and destinations. BPS and WDS show close performance again, even though that BPS also depends on the value of K, while WDS does not. Increasing the number of requests (row eight), MLWDS is the algorithm requiring less state again.

From a global perspective, MLWDS is the algorithm presenting better performance: for the exception of configurations where there is a large number of source AS's and low intensity of requests (rows five and seven), MLWDS requires less

state than any of the other algorithm, which is justified due to second-level aggregation being performed on the sources, which have to keep more state. Hence, for low intensity of requests, performing multi-level aggregation might not be profitable.

BGRP only presents better performance when the number of destinations is low (row five). WDS and BPS show good performance when the intensity of requests is low. However, increasing the intensity of requests impacts significantly in the performance of these two algorithms.

From this specific example, we can infer some preliminary conclusions. First, the intensity of requests is a factor of major importance for the performance of aggregation procedures: any of the algorithms experiences considerable performance variation when the intensity of requests increases. Second, both the proposal of multi-level aggregation based on shared segments, and the sink-tree approach appear to be less sensitive to configurations with high intensity of requests, which seems to indicate that intermediate deaggregation points for the case of first-level aggregation result in a high cost of state along a path. Third, the shared-segment approach requires less amount of state due to aggregates, since it reduces the amount of aggregates generated. Fourth, shared segment second-level aggregation seems a a reasonable solution for inter-domain aggregation, since it combines the insensitivity to the intensity of requests with the minimization of the number of aggregates created.

In the next section, we present simulations that explore in further detail the behavior of BGRP, BPS, WDS, and MLWDS.

V. Performance evaluation

The analytical model presented in section IV demonstrates state accounting by means of a particular scenario, highlighting the impact that different factors have in the overall state of a network. Factors that we considered were the amount of individual requests, number of source and destinations AS's, as well as size of segments shared by requests between their sources and destinations. However, when considering heterogeneous networks such as the Internet, there are other factors that might influence state to be kept, such as traffic distribution, or the average duration of reservations. Hence, to understand the behavior of the algorithms under realistic scenarios, we carry out simulations using the *network simulator version 2* [14].

We model the arrival of requests as a Poisson process[1] with exponential distributed lifetime mean σ, and arrival rate of λ requests per second. To capture the influence of the average duration of requests in the overall state, we use three different types of requests: *short-lived* reservation requests (SLR) with an exponential average duration of 20 s, *long-lived* requests (LLR) with an exponential average duration of 120 s, and a mix of 50% SLR, 50% LLR, that stands for a particular example of mixed traffic (MLR). To distribute the requests across topologies and since there is no current information regarding traffic distribution in the Internet, we apply two different distribution methods: an homogeneous and a *hotspot* method. In the former, source AS's are chosen randomly and destinations are placed according to the distribution of addresses by AS distance mentioned in section IV. In the latter, we use the concept of *hotspot*, i.e., an AS with higher incidence of traffic than the others.

In order to make a consistent comparison of the algorithms, we keep the average number of requests per second in the system (*intensity of requests*) constant, while varying the average duration of requests, according to the traffic intensity formula for an $M/M/\infty$ model [7]. We use this model, since for the case of state accounting, blocking overhead is negligible. For each simulation, state accounting is done both on the AS and edge router level by collecting statistics dynamically for incoming and outgoing reservations: minimum, average and maximum values are updated each time the corresponding variable changes, computed using the formulas presented in Appendix B. With the values obtained, minimum, average, and maximum state per router and per AS are computed with a 95% confidence level. Each simulation experiment has a duration of 1800 s, and data obtained is only considered after an warm-up period of 300 s, to assure that we obtain steady-state results. Also, to achieve statistically meaningful results, each experiment has been repeated several times using different random number seeds.

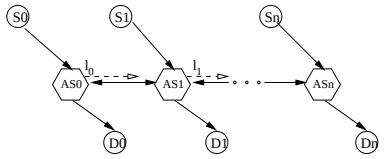


Fig. 9. Generic topology model

[1] We chose a Poisson process to model the arrival of requests since it is known to describe well user *session* arrivals, as mentioned in [15], [13].

A. Simple Flat and Hierarchical Topologies

For the first experiment, we use the topology illustrated in Fig. 9, which was first introduced in the BGRP proposal. This is a generic topology model, useful to model simple but different topologies, that help to show relevant differences in the behavior of the algorithms, according to the function of AS's, i.e., source, destination and transit AS's.

In Fig. 9, each AS k represents a transit AS. S_k represents a group of source AS's and D_k a group of destination AS's directly connected with AS k, $1 < k < n$, $n = 10$. This models a topology where any path has a size less than or equal to ten AS's. Each of the source AS's in S_k sends one reservation to each of the destination AS's in D_k. This model allows to create different kinds of topologies:

- *flat* topologies, where for for each AS k , $\#S_k = \#D_k$;
- *hierarchical* topologies, where $\#S_k$ and $\#D_k$ vary; this property allows to create topologies with more sources and destinations in the edges - *heavy tailed* - or in the interior. So, portions of the topology contribute more than others in terms of requests.

This generic topology is used to model two different scenarios, one that uses a flat topology and one that uses a hierarchical topology, described next.

Flat Topology: This scenario illustrates the behavior of the algorithms in a flat topology where each transit AS k is directly connected to five source and five destination AS's: $\#S_k = \#D_k = 5$, $1 < K < 10$. Each source is sending simultaneously a request for each destination AS, making a total of $50 * 50 = 2500$ requests.

Fig. 12 shows state kept for each of the algorithms in the form of two charts, one that plots state along the path due to individual reservations, both at ingress and egress, and one that plots state along the path due to aggregates, both at ingress and egress. If we look at the upper charts for each algorithm, which plot state due to individual reservations only, we observe that BGRP and MLWDS require the same state, kept only at source and destination AS's. However, BPS and WDS require the same number of units of state at source and destinations, but they also need to keep state about individual reservations at intermediate AS's, which increases the global state. Comparing the lower charts of each algorithm, which plot state due to aggregates only, BGRP is the algorithm that requires more state in terms of aggregates.

These results corroborate that the shared segment approach reduces the number of aggregates created, when compared with the sink-tree approach. However, they do not give any information about the sensitivity of the algorithms to the intensity of requests. Hence, using the same scenario, we increase the intensity of requests in each source of S_k to $Y = 10$ reservations.

In Fig. 13 , looking at the upper charts of each algorithm, each source and destination AS have to keep 500 units of state, for any of the algorithms. BPS and WDS also keep state related with individual reservations at intermediate locations. But, these quantities are higher than the ones presented in Fig. 12 , since the intensity of requests increased. Due to this, there is a significant decay in the performance of BPS and WDS. However, that is not the case for BGRP and MLWDS, which present a similar performance in both scenarios, hence corroborating that both BGRP and MLWDS achieve a fair isolation from the intensity of requests.

Hierarchical Topology: To check if the distribution of requests influences heavily the algorithms, we change the distribution of source and destination AS's in the topology of Fig. . In this scenario, the distribution of S_k and D_k is:

$$\begin{cases} \#S_k = \#D_k = 1, K < 3 \bigvee K > 6 \\ \#S_k = \#D_k = 11, 3 \leq K \leq 6 \end{cases}$$

Thus, there is a total of 50 sources and 50 destinations, sending a total of 2500 requests again, as in the previous scenario. However, in the former all transit AS's would be crossed by an equal number of requests, while in the current scenario more interior AS's are crossed by more requests. Fig. 14 shows again two different charts per algorithm, the upper plotting state due to individual requests and the lower plotting state due to aggregate requests. Regarding state due to individual reservations, BGRP and MLWDS require the same units, since they keep individual requests information only at source and destination AS's. BPS and WDS show quite different results, however. If we look at the charts that plot state due to aggregates, we see that WDS reduces drastically the number of aggregates and that BGRP still creates more aggregates than WDS. The different results show us that any of the algorithms is influenced by the placement of requests across a topology.

As in the previous scenario, we want to weight the sensitivity of the algorithms to the factor intensity of requests under these circumstances. Hence, we increase again the number of requests per source to $Y = 10$. Fig. 14 shows the results,

where we see that any of the algorithms is influenced by the intensity of requests, since state due to individual requests increases proportionally to the intensity of requests. From this experiment, we conclude that the factor intensity of requests has significant influence in state kept and also, that BGRP and MLWDS are the algorithms that show better isolation from this factor.

B. Tree Topology

With this experiment, we aim to highlight the behavior of each algorithm in two extreme cases: a scenario with a large number of destination AS's, unfavourable for BGRP, since there is one source sending several requests to a large number of different destination AS's, and a scenario with one destination AS only, very favourable for BGRP. Hence, we use the particular tree topology illustrated by Fig. 10. Changing the roles of each node yields either a source-tree or a sink-tree scenario.

Source-Tree: To devise the source-tree experiment, a unique source AS, node 0, sends requests to every other node, according to the homogeneous distribution method. In the tree, the maximum path size is five, since this is the current average path size in the Internet.

Fig. 16 outlines state per AS at ingress and egress for each algorithm, when requests are of type SLR and when the intensity of requests is of 5000. If we look at the upper chart of each algorithm we can identify easily the source and destinations: the source keeps state only at egress, while destinations keep state only at ingress. For BPS and WDS we can also spot the intermediate deaggregation points, since they keep state due to individual requests.

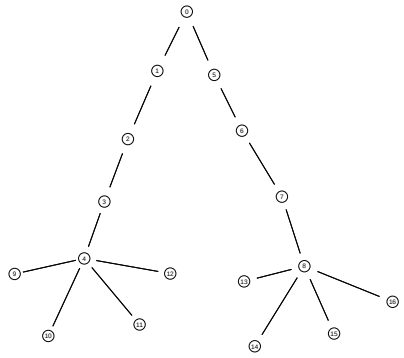


Fig. 10. Tree Topology

Comparing the lower chart of each of the algorithms, we see that BGRP requires in average twice the units of state than the other algorithms for the aggregates it creates. Also, even though MLWDS creates more aggregates at the source than either WDS or BPS, through the remainder path it requires less units of state per AS, due to aggregates. Therefore, from a global perspective, MLWDS is the algorithm that requires less state, because it only deaggregates at the destination, as BGRP, and also because it lowers the number of aggregates created. BPS and WDS also lower the number of aggregates created, but add to this the cost of having to maintain state due to individual reservations at the intermediate deaggregation points, AS 4 and 8: these AS's keep state due to ending aggregates but also due to their mapped individual requests, increasing the global amount of state required.

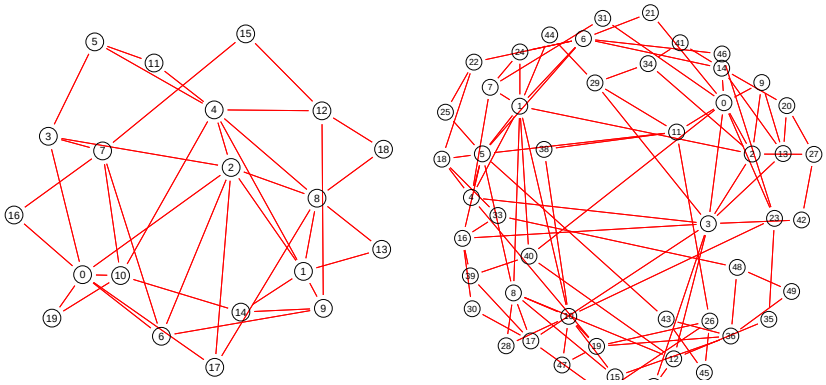
Sink-Tree: Let us now illustrate the behavior of the algorithms in a sink-tree scenario, being the only destination AS represented by node 0. Nodes 1 to 16 represent possible sources requesting reservations to node 0, according to the homogeneous traffic distribution method. Fig. 17 depicts state across the topology for BGRP, BPS, WDS, and MLWDS, when requests are of type SLR and the intensity of requests is 5000. Similarly to the source-tree scenario, we spot sources, destinations and intermediate deaggregation locations by looking at the charts that plot state due to individual reservations for each algorithm, the upper charts.

In terms of aggregates, there is a major difference for this scenario: while BPS still chooses intermediate deaggregation points, both WDS and MLWDS choose as only deaggregation points the destinations. Hence, BGRP, WDS and MLWDS have the same performance. Another major difference from the previous experiment is that for this scenario there is no reduction of aggregates. These results are in compliance with the fact that a sink-tree scenario is a best-case for BGRP.

C. Internet-like Topologies

In this section, we use simulations to investigate the performance of the algorithms on Internet-like topologies. The two topologies used in this section were generated by BRITE [1], a topology generator with the ability to create AS level topologies. First, we devise a scenario where one hotspot AS is placed randomly in a topology. Because we want to assess the impact of having a high intensity of requests either entering or leaving an AS, we devise two specific cases of hotspots: a source and a destination hotspot AS.

Source Hotspot Scenario: The source hotspot scenario is emulated by having 60% of the requests starting in a random hotspot AS in the topology illustrated in Fig. 11 (a). Remaining requests are created by the homogeneous traffic distribution method. This experiment represents a possible source-tree scenario in a realistic environment.



(a) Hotspot topology (b) Homogeneous traffic topology

Fig. 11. Internet-like topologies: (a) is used in the hotspot experiments; (b) is used in the homogeneous traffic experiment

Looking at Tab. VIII, which details state per AS and per router in terms of minimum, average, and maximum values, we verify that request duration seems to influence state kept: MLR requests need the highest amount of state, when using any of the four algorithms, and they also present the largest confidence intervals, showing that there is more state variation. SLR requests demand higher average and maximum state quantities, when compared to LLR requests. A hypothetical explanation for this behavior is the way requests are generated, on the one hand, and the sensitivity of the algorithms to the path size of first requests, on the other: to keep the intensity of requests constant, we generate more SLR requests than either MLR or LLR for the whole simulation duration. Due to the way sources and destinations are placed in the topology, a larger number of requests has more probability of creating more diversified path sizes. Hence, there might be higher probability that the first requests arriving to an AS have different path sizes. This seems to be the case for BPS, which is the most sensitive algorithm to the path size of the first requests arriving. This hypothesis is currently being analysed.

MLWDS shows the best performance for any type of requests: even though that MLWDS creates most likely more aggregates per source, it reduces the average number of aggregates created. WDS, which creates less aggregates, suffers the cost of using several intermediate deaggregation locations. BPS has the lowest performance, also due to the amount of state kept in intermediate deaggregation points. BPS also helds the broadest confidence intervals, which indicate that this algorithm experiences more state variation. However, it should be noticed that BPS also reduces the number of aggregates created when compared with BGRP.

To grasp the influence of the intensity of requests in state, we repeat this experiment for different intensities of requests. The variation of state for different values of σ is plotted in Fig. 18, where each chart represents state variation for a different algorithm. Comparing the charts of the four algorithms, BGRP and MLWDS present similar variation of state units when the intensity of requests changes, even though MLWDS requires slightly less state with the increase of the intensity. Hence, MLWDS performs better for scenarios with high intensity of requests. WDS presents a linear performance decay with the increase of the intensity of requests, because it needs to keep state due to individual requests at intermediate locations. BPS is the algorithm that requires more state, and also the one more affected by the intensity of requests. Also, looking at the values plotted for an intensity of requests of 5000, we attest that BPS shows higher state oscillations than BGRP, WDS or MLWDS for requests of type SLR, which indicates that this algorithm is more sensitive to short-lived requests.

Destination Hotspot Scenario: In this scenario, a node chosen randomly receives 60% of the requests. The remaining 40% requests are placed by using the homogeneous traffic distribution. Tab. IX shows state for this scenario, when the intensity of requests is of 5000. The first evidence presented is that MLWDS achieves the best performance, independently of the duration of requests: the ratio $\frac{MLWDS}{BGRP}$ remains, as in the previous scenario, in the 99%, showing that MLWDS presents an improvement of 1% when compared with BGRP, independently of the duration of requests. The ratios $\frac{BPS}{BGRP}$ and $\frac{WDS}{BGRP}$ show that BPS and WDS experience deterioration up to 30% when compared with BGRP and also, that their performance varies with the duration of requests. This is more visible for BPS, which holds the largest confidence intervals for average values.

TABLE VIII

AVERAGE STATE FOR THE SOURCE HOTSPOT SCENARIO, 5000 REQUESTS.

σ	SCOPE	VARIABLE	BGRP (Avg/ 95% CI)		BPS (Avg / 95% CI)		WDS (Avg/ 95% CI)		MLWDS (Avg/ 95% CI)		$\frac{BPS}{BGRP}$	$\frac{WDS}{BGRP}$	$\frac{MLWDS}{BGRP}$
20s	AS	Min	441.52	437.65, 445.39	663.34	617.55, 709.13	524.86	500.77, 548.95	436.76	432.70, 440.82	1.5	1.19	0.99
(SLR)		Avg	571.51	570.36, 572.66	835.14	788.87, 881.41	684.04	658.19, 709.90	566.16	565.35, 566.96	1.46	1.2	0.99
		Max	704.26	702.64, 705.88	1013.54	963.30, 1063.78	859.15	830.33, 887.97	697.95	696.42, 699.48	1.44	1.22	0.99
	Router	Min	55.19	54.71, 55.67	82.92	77.19, 88.64	65.61	62.60, 68.62	54.60	54.09, 55.10	1.5	1.19	0.99
		Avg	71.44	71.29, 71.58	104.39	98.61, 110.18	85.51	82.27, 88.74	70.77	70.67, 70.87	1.46	1.2	0.99
		Max	88.03	87.83, 88.24	126.69	120.41, 132.97	107.39	103.79, 111.00	87.24	87.05, 87.44	1.44	1.22	0.99
50% 20s	AS	Min	658.88	652.65, 665.11	1000.90	959.64, 1042.16	812.78	729.86, 895.70	653.61	647.15, 660.07	1.52	1.23	0.99
50% 120s		Avg	781.42	778.32, 784.52	1169.33	1125.36, 1213.30	966.67	871.84, 1061.49	776.25	773.12, 779.38	1.5	1.24	0.99
(MLR)		Max	903.52	901.57, 905.47	1336.43	1292.80, 1380.06	1123.54	1017.40, 1229.68	898.31	896.38, 900.24	1.48	1.24	0.99
	Router	Min	82.36	81.58, 83.14	125.11	119.95, 130.27	101.60	91.23, 111.96	81.70	80.89, 82.51	1.52	1.23	0.99
		Avg	97.68	97.29, 98.06	146.17	140.67, 151.66	120.83	108.98, 132.69	97.03	96.64, 97.42	1.5	1.24	0.99
		Max	112.94	112.70, 113.18	167.05	161.60, 172.51	140.44	127.18, 153.71	112.29	112.05, 112.53	1.48	1.24	0.99
120 s	AS	Min	460.87	458.58, 463.16	688.90	660.75, 717.05	539.01	518.24, 559.78	456.30	453.96, 458.64	1.49	1.17	0.99
(LLR)		Avg	564.08	561.07, 567.08	826.98	797.99, 855.96	664.46	641.22, 687.70	559.33	555.93, 562.72	1.47	1.18	0.99
		Max	663.89	661.40, 666.38	960.85	928.82, 992.88	792.69	767.90, 817.48	658.83	656.60, 661.06	1.45	1.19	0.99
	Router	Min	57.61	57.32, 57.89	86.11	82.59, 89.63	67.38	64.78, 69.97	57.04	56.74, 57.33	1.49	1.17	0.99
		Avg	70.51	70.13, 70.89	103.37	99.75, 107.00	83.06	80.15, 85.96	69.92	69.49, 70.34	1.47	1.18	0.99
		Max	82.99	82.68, 83.30	120.11	116.10, 124.11	99.09	95.99, 102.19	82.35	82.08, 82.63	1.45	1.19	0.99

In terms of different duration of requests, results for this scenario are similar to the ones obtained in the previous experiment. MLWDS shows again the best performance for any type of requests; BGRP presents a close performance to MLWDS; BPS has the lowest performance.

TABLE IX

AVERAGE STATE FOR THE DESTINATION HOTSPOT SCENARIO, 5000 REQUESTS.

σ	SCOPE	VARIABLE	BGRP (Avg/ 95% CI)		BPS (Avg/ 95% CI)		WDS (Avg/ 95% CI)		MLWDS (Avg/ 95% CI)		$\frac{BPS}{BGRP}$	$\frac{WDS}{BGRP}$	$\frac{MLWDS}{BGRP}$
20s	AS	Min	443.29	438.07, 448.51	601.79	442.10, 761.48	497.27	474.85, 519.69	437.86	432.74, 442.98	1.36	1.12	0.99
(SLR)		Avg	570.94	567.82, 574.05	760.21	589.74, 930.68	648.19	624.13, 672.25	566.23	563.64, 568.82	1.33	1.14	0.99
		Max	702.99	699.32, 706.66	930.69	747.74, 1113.64	817.13	788.42, 845.84	698.66	695.45, 701.87	1.32	1.16	0.99
	Router	Min	55.41	54.76, 56.06	75.22	55.26, 95.18	62.16	59.36, 64.96	54.73	54.09, 55.37	1.36	1.12	0.99
		Avg	71.37	70.98, 71.76	95.03	73.72, 116.34	81.02	78.02, 84.03	70.78	70.46, 71.10	1.33	1.14	0.99
		Max	87.87	87.41, 88.33	116.34	93.47, 139.21	102.14	98.55, 105.73	87.33	86.93, 87.73	1.32	1.16	0.99
50% 20s	AS	Min	651.51	643.86, 659.16	799.28	607.99, 990.57	750.82	713.96, 787.68	646.98	639.60, 654.36	1.23	1.15	0.99
50% 120s		Avg	778.46	774.91, 782.01	955.31	747.90, 1162.71	902.55	854.20, 950.89	773.73	770.29, 777.16	1.23	1.16	0.99
(MLR)		Max	897.04	890.67, 903.41	1103.40	889.17, 1317.63	1051.24	993.18, 1109.30	892.51	886.13, 898.89	1.23	1.17	0.99
	Router	Min	81.44	80.48, 82.40	99.91	76.00, 123.82	93.85	89.24, 98.46	80.87	79.95, 81.79	1.23	1.15	0.99
		Avg	97.31	96.86, 97.75	119.41	93.49, 145.34	112.82	106.78, 118.86	96.72	96.29, 97.14	1.23	1.16	0.99
		Max	112.13	111.33, 112.93	137.93	111.15, 164.70	131.41	124.15, 138.66	111.56	110.77, 112.36	1.23	1.17	0.99
120 s	AS	Min	459.40	454.09, 464.71	612.98	496.14, 729.82	523.55	503.62, 543.48	454.38	449.10, 459.66	1.33	1.14	0.99
(LLR)		Avg	564.98	561.97, 567.98	746.63	620.24, 873.02	649.22	627.38, 671.06	559.87	556.71, 563.04	1.32	1.15	0.99
		Max	665.53	662.86, 668.20	876.43	745.72, 1007.14	775.39	750.16, 800.62	660.69	657.82, 663.56	1.32	1.17	0.99
	Router	Min	57.43	56.76, 58.09	76.62	62.02, 91.23	65.44	62.95, 67.93	56.80	56.14, 57.46	1.33	1.14	0.99
		Avg	70.62	70.25, 71.00	93.33	77.53, 109.13	81.15	78.42, 83.88	69.98	69.59, 70.38	1.32	1.15	0.99
		Max	83.19	82.86, 83.52	109.55	93.21, 125.89	96.92	93.77, 100.08	82.59	82.23, 82.95	1.32	1.17	0.99

We repeated the simulation while varying the intensity of requests and Fig. 19 plots the results obtained, where once again the pattern of behavior of BGRP and MLWDS is similar, corroborating the values detailed in Tab. IX. In comparison to the previous scenario, there is a decrease of average state for WDS and BPS, while BGRP requires approximately the same state. For instance, when the intensity of requests is of 5000, both MLWDS and BGRP require approximately 800 state units for the source and for the destination hotspot scenario, when requests are of type MLR. When requests are either of type SLR or LLR, both MLWDS and BGRP require approximately 600 state units. In contrast, both BPS and WDS, require more state units for the source-tree scenario than for the sink-tree scenario.

D. Homogeneous Traffic

The hotspot experiments allow us to conclude that having an AS with higher incidence of requests impacts the performance of the algorithms. However, we cannot assess the effect of having an AS with high incidence of requests in the overall state, without further evaluating the algorithms in scenarios where there are no hotspots. Hence, the next experiment exemplifies BGRP, BPS, WDS, and MLWDS behavior in a topology with no hotspots. We use a larger AS level topology, illustrated in Fig. 11 (b) , where requests are distributed by the homogeneous method.

Tab. X details state per AS and per router in terms of minimum, average, and maximum for the three algorithms, when the intensity of requests is of 5000. A major difference from previous scenarios is that state kept per AS and per router is lower, due to the fact that the current topology is bigger than the topology used in previous experiments in this section.

Results obtained in the previous experiments are coherent with the results shown in Tab. X. In general terms, MLR requests still require more state for any of the algorithms; MLWDS still presents the best performance. BPS and WDS show a performance decay up to 30% when compared with either MLWDS or BGRP, and BPS achieves the poorest performance.

TABLE X

AVERAGE STATE FOR THE HOMOGENEOUS TRAFFIC SCENARIO, 5000 REQUESTS.

σ	SCOPE	VARIABLE	BGRP (Avg/95% CI)	BPS (Avg/95% CI)	WDS (Avg/95% CI)	WDS (Avg/95% CI)	$\frac{BPS}{BGRP}$	$\frac{WDS}{BGRP}$	$\frac{MLWDS}{BGRP}$
20s	AS	Min	242.10 241.44, 242.76	322.87 315.60, 330.14	267.87 266.65, 269.10	237.84 236.33, 239.36	1.33	1.11	0.98
(SLR)		Avg	373.32 372.71, 373.94	479.80 471.62, 487.99	415.80 415.50, 416.11	385.92 382.67, 389.16	1.29	1.11	1.03
		Max	506.21 504.77, 507.66	660.60 650.62, 670.59	569.67 567.48, 571.86	539.98 533.07, 546.90	1.30	1.13	1.07
	Router	Min	30.26 30.18, 30.34	40.36 39.45, 41.27	33.48 33.33, 33.64	29.73 29.54, 29.92	1.33	1.11	0.98
		Avg	46.67 46.59, 46.74	59.98 58.95, 61.00	51.98 51.94, 52.01	48.24 47.83, 48.64	1.29	1.11	1.03
		Max	63.28 63.10, 63.46	82.58 81.33, 83.82	71.21 70.93, 71.48	67.50 66.63, 68.36	1.30	1.13	1.07
50% 20s	AS	Min	350.00 347.74, 352.25	478.76 469.41, 488.11	422.21 418.82, 425.60	334.54 331.81, 337.26	1.37	1.21	0.96
50% 120s		Avg	460.30 459.00, 461.59	622.20 613.45, 630.96	555.62 553.36, 557.88	446.60 444.20, 449.00	1.35	1.21	0.97
(SLR)		Max	567.41 566.04, 568.78	775.35 766.40, 784.29	688.00 684.86, 691.14	557.09 554.26, 559.93	1.37	1.21	0.98
	Router	Min	43.75 43.47, 44.03	59.85 58.68, 61.01	52.78 52.35, 53.20	41.82 41.48, 42.16	1.37	1.21	0.96
		Avg	57.54 57.38, 57.70	77.78 76.68, 78.87	69.45 69.17, 69.74	55.83 55.53, 56.13	1.35	1.21	0.97
		Max	70.93 70.76, 71.10	96.92 95.80, 98.04	86.00 85.61, 86.39	69.64 69.28, 69.99	1.37	1.21	0.98
120 s	AS	Min	266.31 264.42, 268.20	337.17 328.87, 345.46	296.24 293.09, 299.38	251.63 250.13, 253.13	1.27	1.11	0.94
(LLR)		Avg	363.54 362.21, 364.87	456.16 448.87, 463.44	408.54 406.50, 410.58	351.22 349.83, 352.60	1.25	1.12	0.97
		Max	459.80 457.75, 461.85	586.40 579.23, 593.58	522.27 519.26, 525.28	451.28 449.01, 453.56	1.28	1.14	0.98
	Router	Min	33.29 33.05, 33.52	42.15 41.11, 43.18	37.03 36.64, 37.42	31.45 31.27, 31.64	1.27	1.11	0.94
		Avg	45.44 45.28, 45.61	57.02 56.11, 57.93	51.07 50.81, 51.32	43.90 43.73, 44.07	1.25	1.12	0.97
		Max	57.48 57.22, 57.73	73.30 72.40, 74.20	65.28 64.91, 65.66	56.41 56.13, 56.69	1.28	1.14	0.98

These results still hold when varying the intensity of requests, as shown in Fig. 20. BGRP and MLWDS show close performance, being MLWDS the algorithm that requires less global state. WDS and BPS performance is similar, even though that BPS requires more state than WDS.

From the results obtained, we conclude that MLWDS achieves the best performance for any of the experiments. Both BGRP and MLWDS present good insensitivity to the intensity of requests. In contrast, both BPS and WDS performance experiences significant decay due to the increase of the intensity of requests. Also, even though that any of the algorithms shows state variation for different duration of requests, this variation is more noticeable for BPS. Our hypothetical justification to this behavior, still under study, is the higher sensitivity of BPS to the path size of the first requests arriving to an AS.

VI. SUMMARY AND CONCLUSIONS

In this paper we investigated different aggregation approaches for inter-domain control aggregation. Our goal was to gain greater insight into the ability of different inter-domain aggregation procedures in accommodating large volumes of reservation requests across different routing domains. As utility function, we considered the minimization of state to be kept per AS and per edge router. We evaluated two basic aggregation approaches, sink-tree and shared segment based, and four algorithms derived from these approaches. BGRP follows the sink-tree approach, and BPS, WDS, and MLWDS follow the shared path segment approach. We examined state accounting by means of a simple analytical example and also by means of simulations. For each simulation experiment, we varied the number of AS's, duration of requests, distribution

of requests through time, and also distribution of requests across the topology, so we could assess the impact of each of these factors on the different aggregation methods under consideration.

The analytical example attested that the intensity of individual reservation requests is of major importance for any of the approaches. However, it is more relevant for the shared segment approach in terms of first-level aggregation, due to the additional intermediate deaggregation points this approach introduces: with first-level aggregation, intermediate deaggregation points need to maintain information not only about aggregates, but also about the individual requests that are mapped to the deaggregated aggregates. However, this cost can be avoided if we perform instead second-level aggregation, as suggested by using the algorithm MLWDS.

Simulations using different topologies corroborated that MLWDS achieves better performance for the scenarios presented. Results also show that MLWDS performance improves with the increase of the intensity of requests. WDS and BPS, the two algorithms that perform first-level aggregation only and that are based on the shared segment approach show deterioration in their performance, due to the cost of intermediate deaggregation points.

A first conclusion to draw from this investigation is that by performing second-level aggregation, the shared-segment approach achieves better performance in terms of minimization of state maintained along a path, when compared with the sink-tree approach. The implementation of this mechanism is of reasonable complexity, since only the sources perform second-level aggregation, and its sensitivity to the intensity of requests is low. However, it is also necessary to verify its complexity in terms of signaling load.

A second conclusion is that the sink-tree approach also represents a reasonable solution in terms of minimizing the amount of state maintained. It is also of reasonable complexity and of low sensitivity to the number of requests. However, its one disadvantage is that it is not optimal in terms of the number of aggregates it creates. Algorithms based on the shared segment approach reduce the number of aggregates when compared with BGRP.

As future work, we are investigating whether the use of multi-level aggregation can lower the sensitivity of the shared-segment approach to traffic intensity and also, its complexity and drawbacks. We are also devising an algorithm based on the shared segment approach that avoids keeping state of individual reservations in intermediate deaggregation locations, and we are investigating the bandwidth efficiency and overall signaling load of these different approaches. Both are important performance measures beyond the amount of reservation state, that was the focus of this paper.

REFERENCES

[1] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byersers. BRITE: An Approach to Universal Topology Generation. In *MASCOTS '01*, August 2001.

[2] Anja Feldmann, Anna C. Gilbert, Polly Huang, and Walter Willinger. Dynamics of IP traffic: A study of the role of variability and the impact of control. *SIGCOMM'99*, 1999.

[3] Bob Braden, David Clark, and Scott Shenker. Integrated Services in the Internet Architecture: an Overview. *Request for Comments 1633, Internet Engineering Task Force*, June 1994.

[4] Bob Braden, Lixia Zhang, and Sugih Jamin. Resource Reservation Protocol (RSVP) - version 1, Functional Specification. *Request for Comments 2205, Internet Engineering Task Force*, September 1997.

[5] Luca Degrossi and Louis Berger. Internet Stream Protocol Version 2 (ST2). *Request for Comments 1819, Internet Engineering Task Force*, August 1995.

[6] Ping Pan, Ellen Hahne, and Henning Schulzrinne. The Border Gateway Reservation Protocol (BGRP) for Tree-Based Aggregation of Inter-Domain Reservations. *Journal of Communications and Networks*, June 2000.

[7] Raj Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, Inc., 1991.

[8] Roch Guérin, Shai Herzog, and Stephen Blake. Aggregating RSVP-Based QoS Requests. *Internet Draft, Internet Enginnering Task Force*, September 1997. Work in Progress.

[9] Steve Uhling and Olivier Bonaventure. Implications of Interdomain Traffic Characteristics on Traffic Enginering. Technical report, University of Namur, June 2001.

[10] Steven Berson and Subramaniam Vincent. Aggregation of Internet Integrated Services State. *Internet Draft, Internet Engineering Task Force*, August 1998. Work in Progress.

[11] Steven Blake, David Black, Mark Carlson, Elwyn Davies, Zheng Wang, and Walter Weiss. An Architecture for Differentiated Services. *Request for Comments 2475, Internet Engineering Task Force*, December 1998.

[12] Telstra. BGP Table Report. http://bgp.potaroo.net/, February 2001.

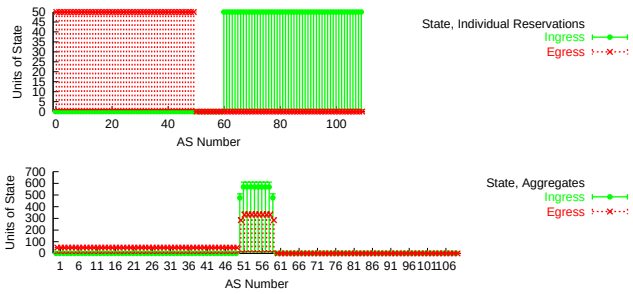
[13] Vern Paxson and Sally Floyd. Why We Don't Know How to Simulate the Internet. *Winter Simulation Conference*, 1997.

[14] VINT Project. *The ns Manual*. UC Berkeley, LBL, USC/ISI, Xerox Parc, September 2001.

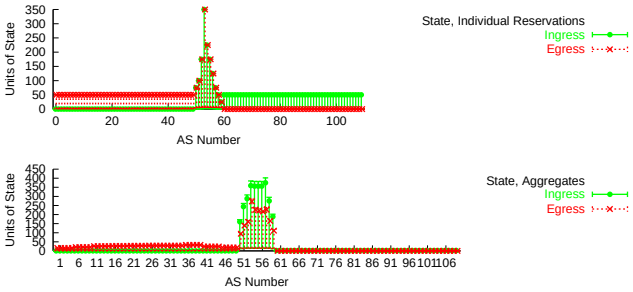
[15] Walter Willinger and Vern Paxson. Where Mathematics meets the Internet. *Notices of the American Mathematical Society, 45(8)*, August 1998.

[16] Yakov Rekhter and Tony Li. A Border Gateway Protocol 4 (BGP-4). *Request for Comments 1771, Internet Engineering Task Force*, March 1995.

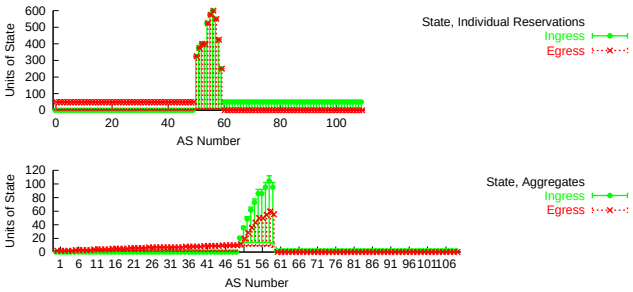
APPENDIX A: RESULTS



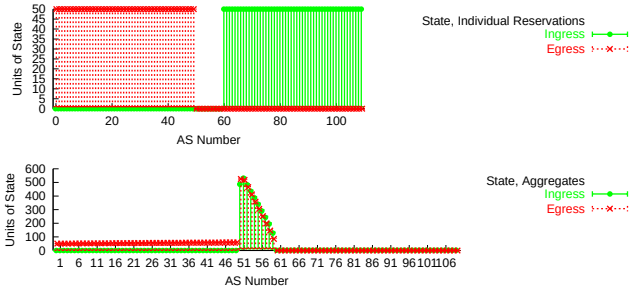
(a) BGRP



(b) BPS

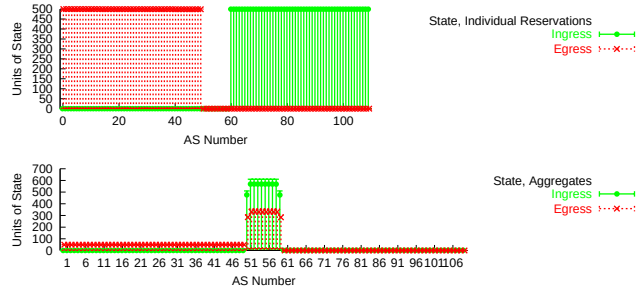


(c) WDS

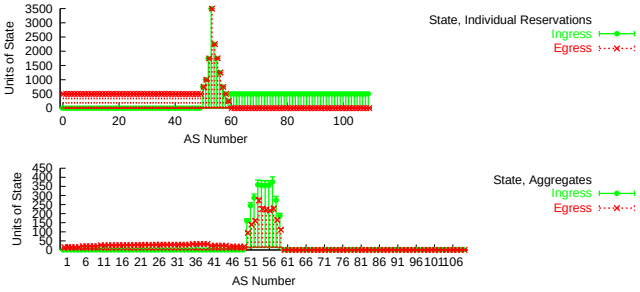


(d) MLWDS

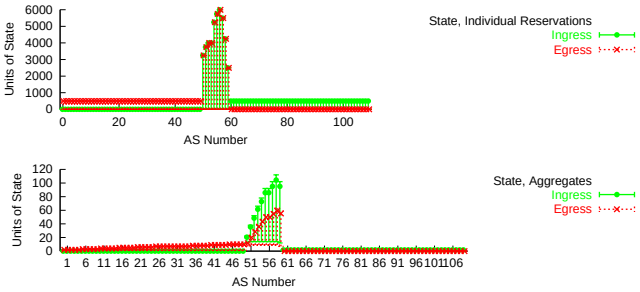
Fig. 12. Global state in a flat topology, $Y = 1$



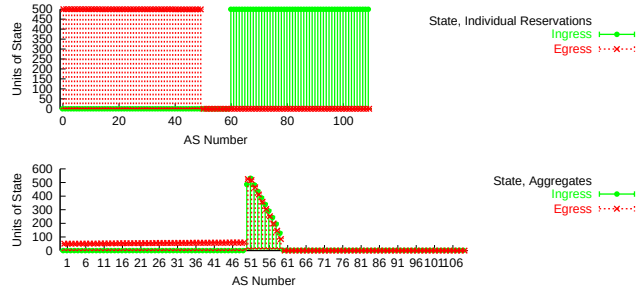
(a) BGRP



(b) BPS

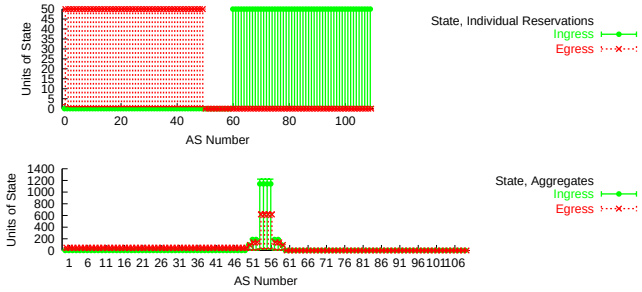


(c) WDS

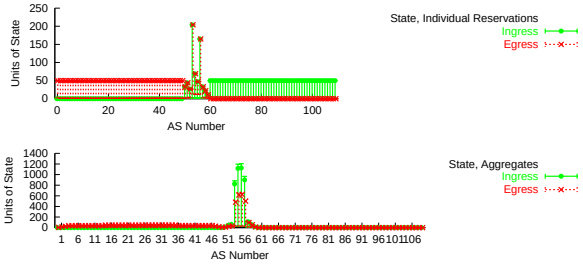


(d) MLWDS

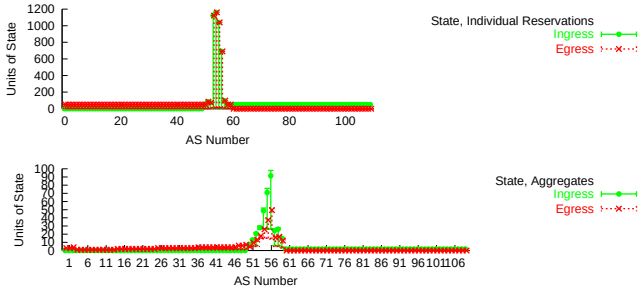
Fig. 13. Global state in a flat topology, $Y = 10$



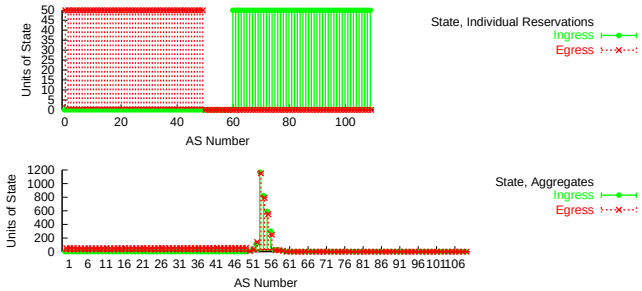
(a) BGRP



(b) BPS

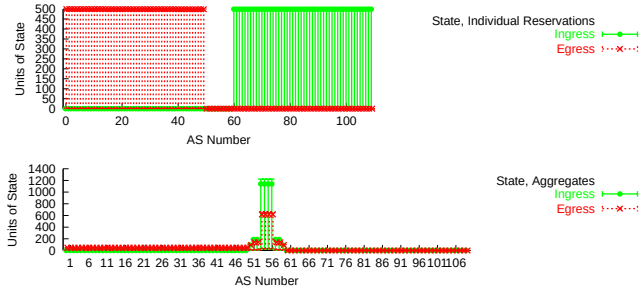


(c) WDS

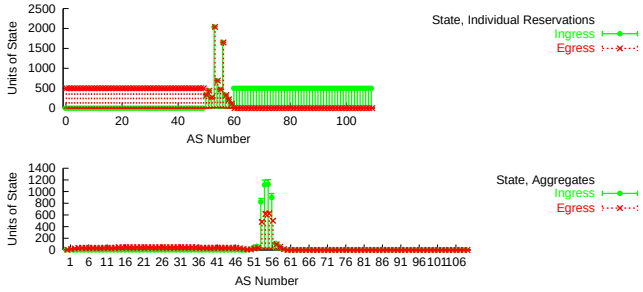


(d) MLWDS

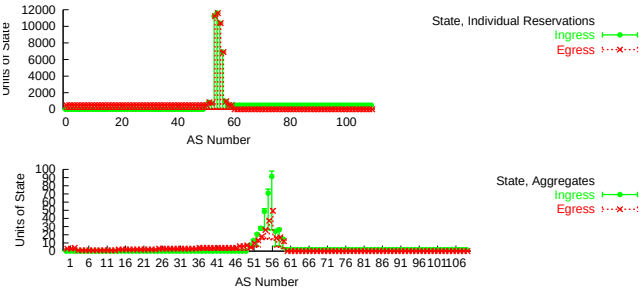
Fig. 14. Global state in a hierarchical topology, $Y = 1$



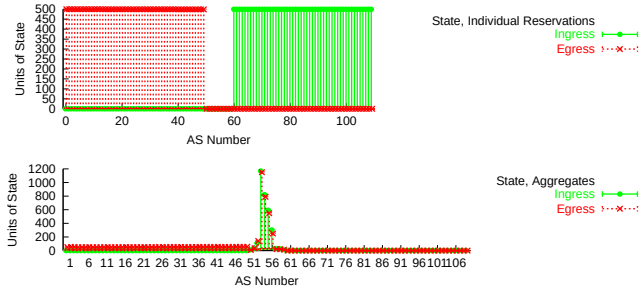
(a) BGRP



(b) BPS

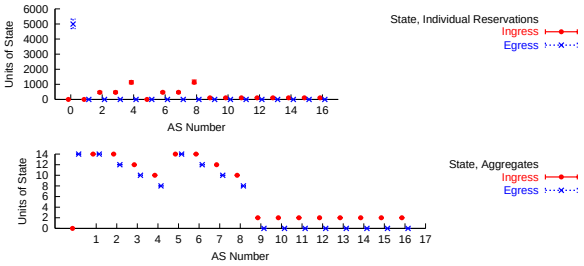


(c) WDS

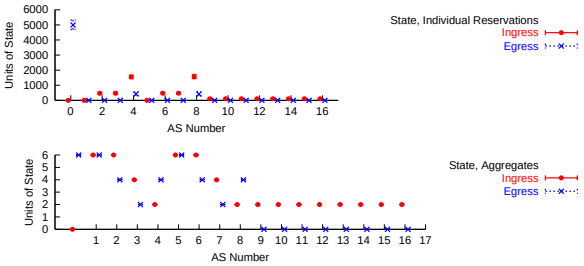


(d) MLWDS

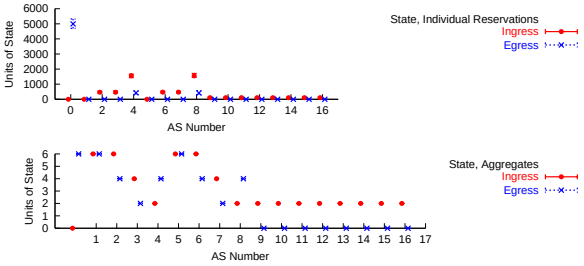
Fig. 15. Global state in a hierarchical topology, $Y = 10$



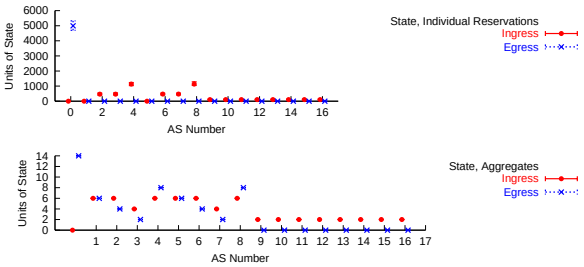
(a) BGRP



(b) BPS

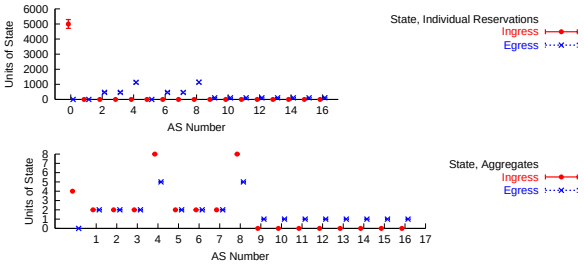


(c) WDS

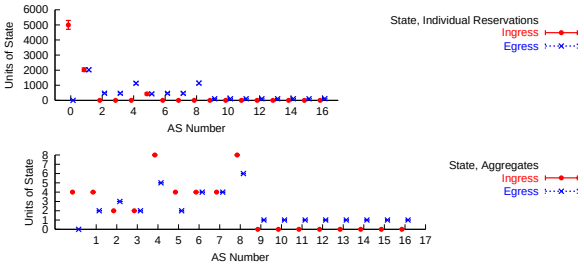


(d) MLWDS

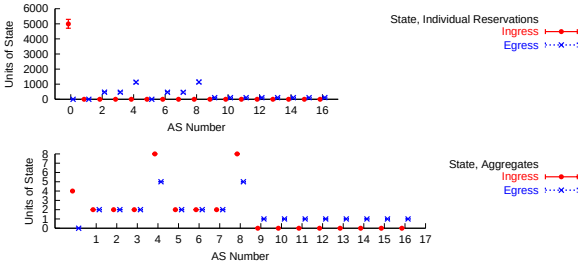
Fig. 16. Global state in a source-tree, $\sigma = 20s$, 5000 requests



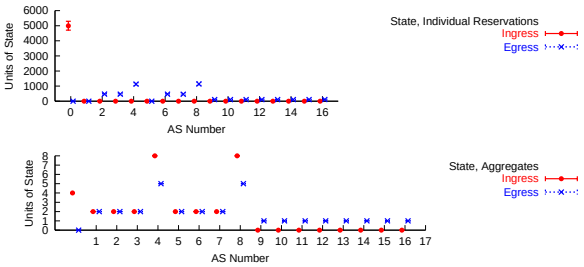
(a) BGRP



(b) BPS

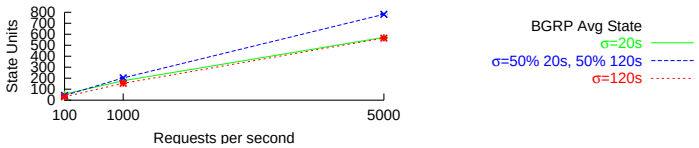


(c) WDS

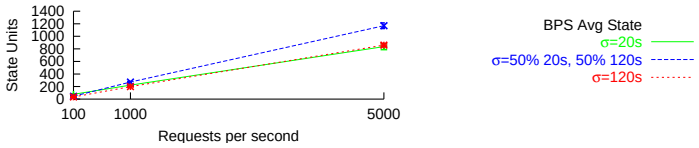


(d) MLWDS

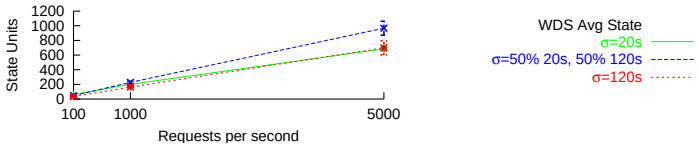
Fig. 17. Global state in a sink-tree, $\sigma = 20s$, 5000 requests



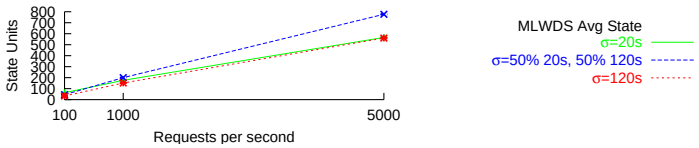
(a) BGRP



(b) BPS

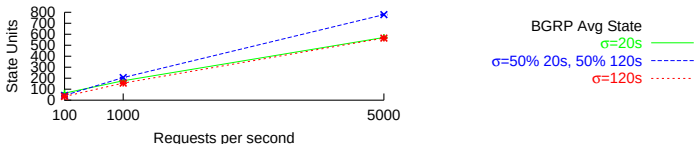


(c) WDS

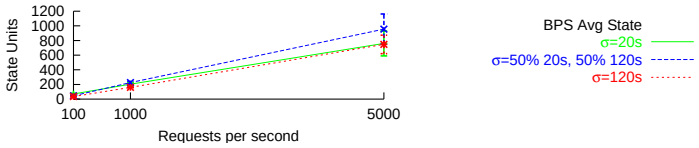


(d) MLWDS

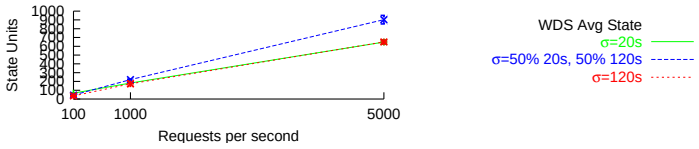
Fig. 18. State variation, source hotspot scenario



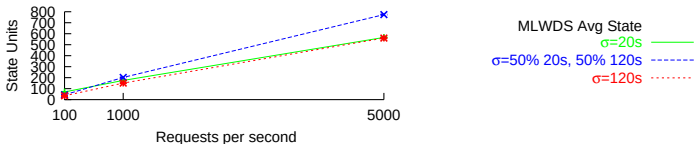
(a) BGRP



(b) BPS

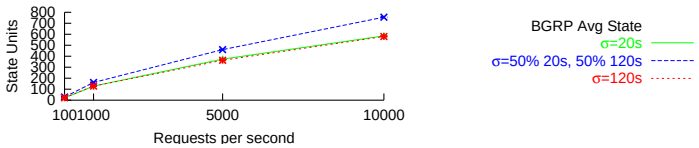


(c) WDS

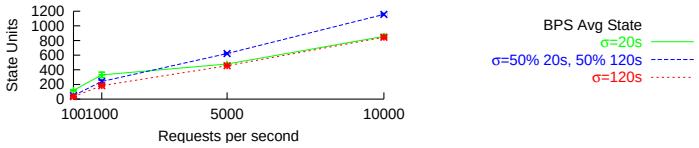


(d) MLWDS

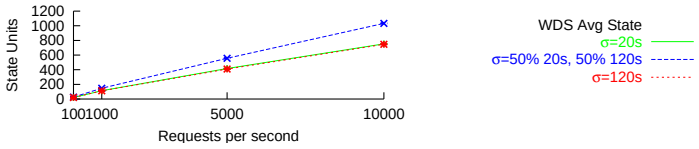
Fig. 19. State variation, destination hotspot scenario



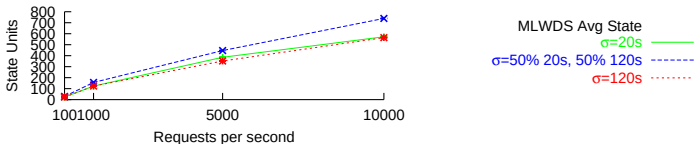
(a) BGRP



(b) BPS



(c) WDS



(d) MLWDS

Fig. 20. State variation, homogeneous traffic scenario

APPENDIX B: FORMULAS

Formulas presented in this appendix were used to calculate the maximum, average and minimum values for each of the variables used in the simulations (Section 4). N represents a variable, r an edge router and X an AS. For instance, $Average_r^N$ represents the average of variable N at router r . t_{stop} represents the instant when the simulation ended. t_{warmup} represents the instant after an adequate warmup period.t_i represents an instant when N suffered a change. n represents the number of edge routers at AS X and a represents the total number of AS's for a specific scenario.

Statistic per Router

- N Average at router r: $Average_r^N = \frac{\sum_{i=0}^{n} N_i(t_{i+1}-t_i)}{t_{stop}-t_{warmup}}, t_i \in [t_{warmup}, t_{stop}]$
- N Maximum at router r: $MAX_r^N = max(N_{t_i}, N_{t_{i-1}}, ...), t_i \in [t_{warmup}, t_{stop}]$
- N Minimum at router r: $MIN_r^N = min(N_{t_i}, N_{t_{i-1}}, ...), t_i \in [t_{warmup}, t_{stop}]$

Statistic per AS

- N Average at AS X: $Average_X^N = \sum_{i=0}^{n} Average_{r_i}^N, i < n$
- N Maximum at AS X: $MAX_r^N = \sum_{i=0}^{n} MAX_{r_i}^N, i < n$
- N Minimum at AS X: $MIN_r^N = \sum_{i=0}^{n} MIN_{r_i}^N, i < n$

Global Statistic

- Global Average: $Average_N = \frac{\sum_{i=0}^{n} Average_{x_i}^N}{a}$
- Global Variance: $var_N = \frac{\sum_{i=0}^{n} (Average_{x_i}^N - Average_N)^2}{a}$
- Global Standard Deviation: $sd_N = \sqrt{var} = \sqrt{\frac{\sum_{i=0}^{n} (Average_{x_i}^N - Average_N)^2}{a}}$